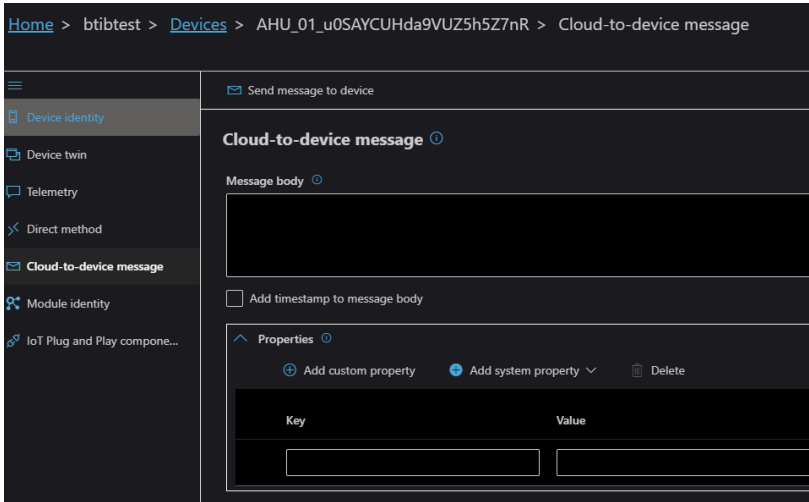


Step 5 Send commands from IoT Hub to Niagara

IoT Hub Connector supports cloud to niagara communication via IoT Hub C2D messages to control points remotely to do this.

Send a point action command

1. Go to the **Azure IoT Explorer** application then in your IoT Hub devices then click on the device where you want to send messages. Tehn cloud to device messages.



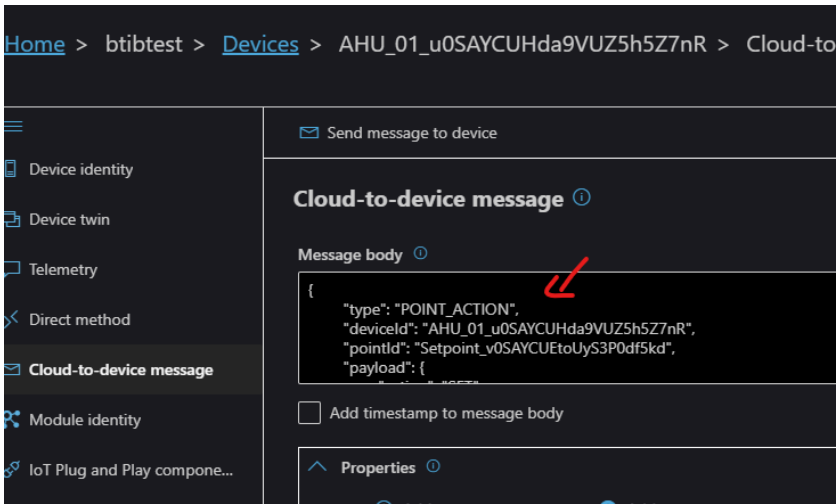
2. By default we use this message template for **POINT_ACTION** command. by you can use any format that meat your needs. check the connector advanced settings.
 - a. This is the default command template.

```
{
  "type": "POINT_ACTION",
  "deviceId": "AHU_01_u0SAYCUHda9VUZ5h5Z7nR",
  "pointId": "Setpoint_v0SAYCUEtoUyS3P0df5kd",
  "payload": {
    "action": "SET",
    "value": 1111
  }
}
```

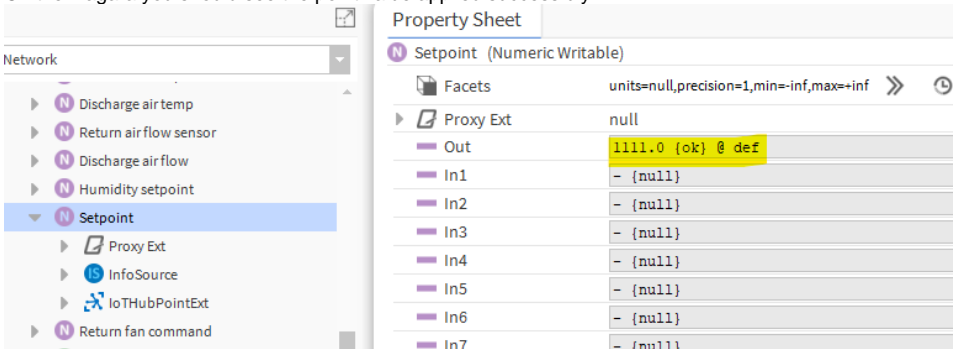
- b. This is the default connector point action command configuration.

Commands Policy	Single Point Command
Message Type	{json('type')}
Command Set Object	{json('').escape}
Command Device Id	{json('deviceId')}
Command Point Id	{json('pointId')}
Command Action	{json('payload.action')}
Command Value	{json('payload.value')}
Command Duration	{json('payload.duration')}

3. On the left chose Cloud To Device Message.



4. Then hit **Send Message**.
5. On the niagara you should see the point value applied successfully.

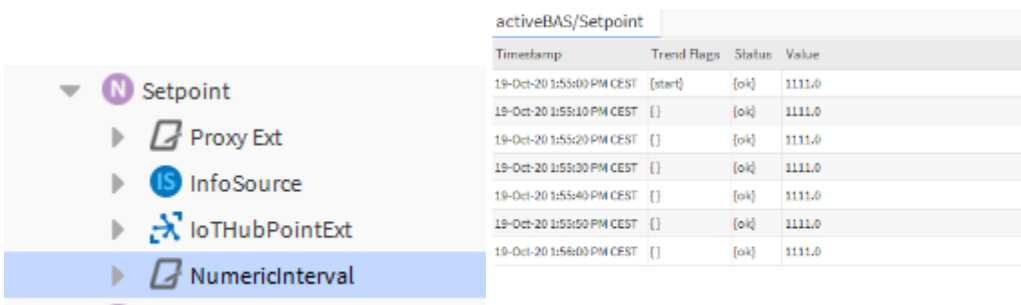


6. And the new value sent to the cloud.

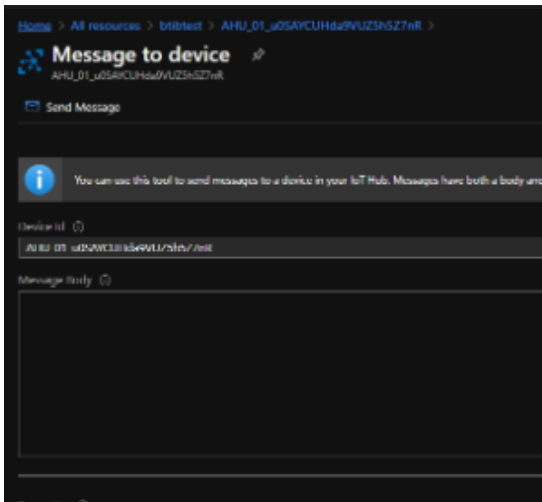
Send a send history command

You can pull historical value for any point that has a history associated by sending a **SEND_HISTORY** command.

1. Add a history. extension to the point.



2. Go to the portal and send the command to the device.



3. By default we use this message template for send history command. by you can use any format that meat your needs. check the connector advanced settings.

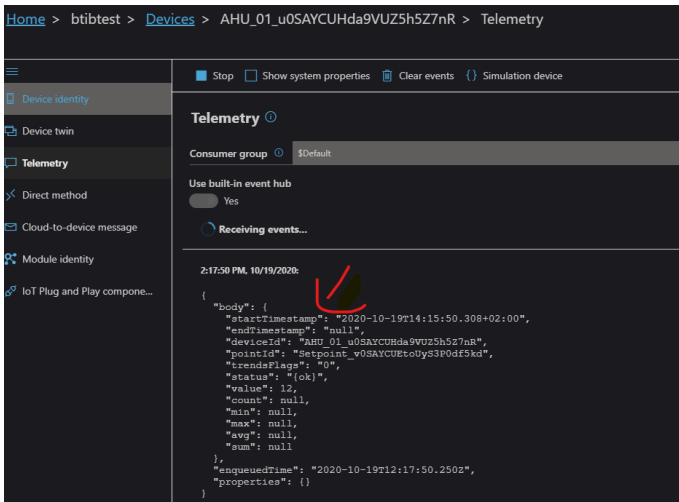
a. This is the default command template.

```
{
  "type": "SEND_HISTORY",
  "deviceId": "AHU_01_u0SAYCUHda9VUZ5h5Z7nR",
  "pointId": "Setpoint_v0SAYCUEtoUyS3P0df5kd",
  "payload": {
    "startDate": "2020-10-19T13:15:55.011+02:00",
    "endDate": "2020-10-19T14:15:55.011+02:00"
  }
}
```

b. This is the default connector send history command configuration.

Commands Policy	Single Point Command	
Message Type	<code>{json('type')}</code>	?
Command Set Object	<code>{json('').escape}</code>	?
Command Device Id	<code>{json('deviceId')}</code>	?
Command Point Id	<code>{json('pointId')}</code>	?
Command Action	<code>{json('payload.action')}</code>	?
Command Value	<code>{json('payload.value')}</code>	?
Command Duration	<code>{json('payload.duration')}</code>	?
Start Date	<code>{json('payload.startDate')}</code>	?
End Date	<code>{json('payload.endDate')}</code>	?
Delta	<code>{json('payload.delta')}</code>	?
Roll Up	<code>{json('payload.rollup')}</code>	?

4. And you should see the messages being sent.



To change the message format check the connector advanced setting then history message template

History Message Variables	<pre> \$(startTimestamp) \$(endTimestamp) \$(pointId) \$(deviceId) \$(status) \$(value) \$(trendsFlags) \$(count) \$(min) \$(max) \$(avg) \$(sum) </pre>
History Message Template	<pre> { "startTimestamp": "\$(startTimestamp)", "endTimestamp": "\$(endTimestamp)", "deviceId": "\$(deviceId)", "pointId": "\$(pointId)", "trendsFlags": "\$(trendsFlags)", "status": "\$(status)", "value": \$(value), "count": \$(count), "min": \$(min). </pre>

Send ack alarm command

You can ack alarms by sending an **ACK_ALARM** command to any alarm recipient device.

- By default we use this message template for ack alarm command. by you can use any format that meat your needs. check the connector advanced settings.
 - This is the default command template.

```

{
  "type": "ACK_ALARM",
  "payload": {
    "uuid": "85a1fe08-4568-428c-ae10-e4d35f861dc3"
  }
}

```

- This is the default connector send history command configuration.

Alarm Uuid	<pre> "min": \$(min), {json('payload.uuid')} </pre>
------------	---

- Go the alarms console and pick an unacked alarm id.

Alarm History				
Timestamp	Source State	Ack State	Source	Alarm
19-Oct-20 10:35:36 AM CEST	Offnormal	Unacked	slot:/tesFlex9/points/iotCoreGoogle/GoogleIoTCorePointTestExt	Defa
19-Oct-20 10:35:36 AM CEST				
19-Oct-20 10:35:36 AM CEST				
19-Oct-20 10:35:10 AM CEST				
19-Oct-20 10:35:10 AM CEST				
19-Oct-20 10:35:10 AM CEST				
19-Oct-20 10:35:09 AM CEST				

Alarm Record

Timestamp	19-Oct-20 10:35:36 AM CEST
Uuid	85a1fe08-4568-428c-ae10-e4d35f861dc3
Source State	Offnormal
Ack State	Unacked
Ack Required	true

3. Go to device and send the command.

[Home](#) >
 [btibttest](#) >
 [Devices](#) >
 alarms >
 Cloud-to-device message

Device identity

Device twin

Telemetry

Direct method

Cloud-to-device message

Module identity

IoT Plug and Play compone...

Send message to device

Cloud-to-device message ⓘ

Message body ⓘ

```
{
  "type": "ACK_ALARM",
  "payload": {
    "uuid": "85a1fe08-4568-428c-ae10-e4d35f861dc3"
  }
}
```

☐ Add timestamp to message body

Properties ⓘ

4. Go back to the console and you should see that the alarm has been acked.

Alarm History				
Timestamp	Source State	Ack State	Source	Alarm Class
19-Oct-20 10:35:36 AM CEST	Offnormal	Acked	slot:/tesFlex9/points/iotCoreGoogle/GoogleIoTCorePointTestExt	Default Alarm
19-Oct-20 10:35:36 AM CEST	Offnormal	Unacked	slot:/tesFlex8/points/iotCoreGoogle/GoogleIoTCorePointTestExt	Default Alarm
19-Oct-20 10:35:36 AM CEST				
19-Oct-20 10:35:10 AM CEST				
19-Oct-20 10:35:10 AM CEST				
19-Oct-20 10:35:10 AM CEST				

Alarm Record

Timestamp	19-Oct-20 10:35:36 AM CEST
Uuid	85a1fe08-4568-428c-ae10-e4d35f861dc3
Source State	Offnormal

Next Step

[Step 6 Consume Data from IoT Hub in Azure](#)